Bilkent University

Computer Science Department

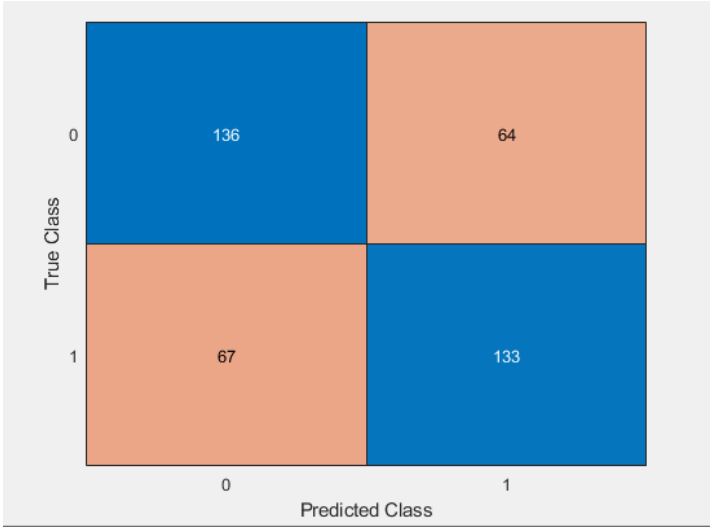
CS464 Introduction to Machine Learning
Spring 2020
Homework 2

Muhammed Musab OKŞAŞ, 21602984

Question 1.1 and 1.2

Performance Metrics for mini batch Hog dataset

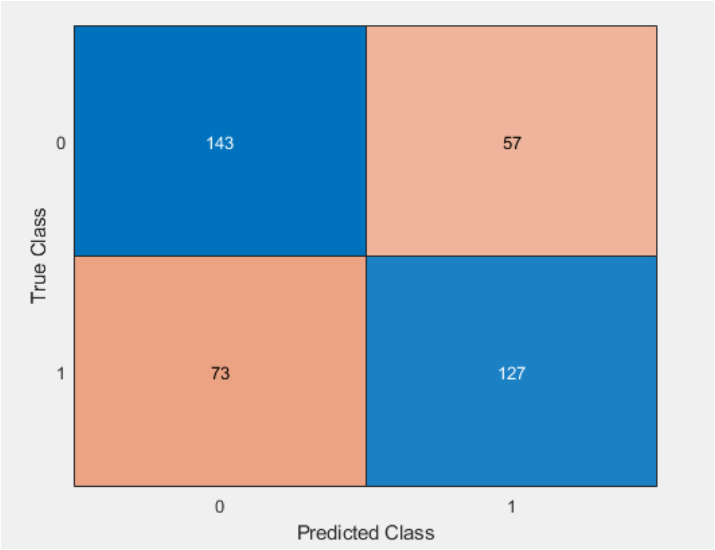
Training time: **4.11** seconds, Confusion Matrix:



tp: 133, tn: 136, fp: 64, fn: 67, accuracy: 0.672500
precision: 0.68, recall: 0.67, specificity: 0.68, negative predictive value: 0.67
false positive rate: 0.32, false discovery rate: 0.32, f1: 0.67, f2: 0.67

Performance Metrics for full batch Hog dataset

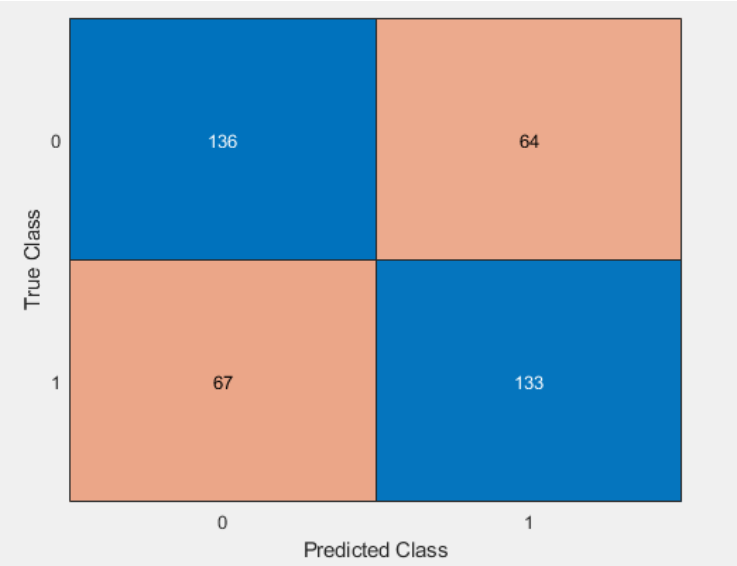
Training time: **0.6** seconds, Confusion Matrix:



tp: 127, tn: 143, fp: 57, fn: 73, accuracy: 0.675000
precision: 0.69, recall: 0.64, specificity: 0.71, negative predictive value: 0.66
false positive rate: 0.28, false discovery rate: 0.31, f1: 0.66, f2: 0.65

Performance Metrics for stochastic Hog dataset

Training time: **6.6** seconds, Confusion Matrix:



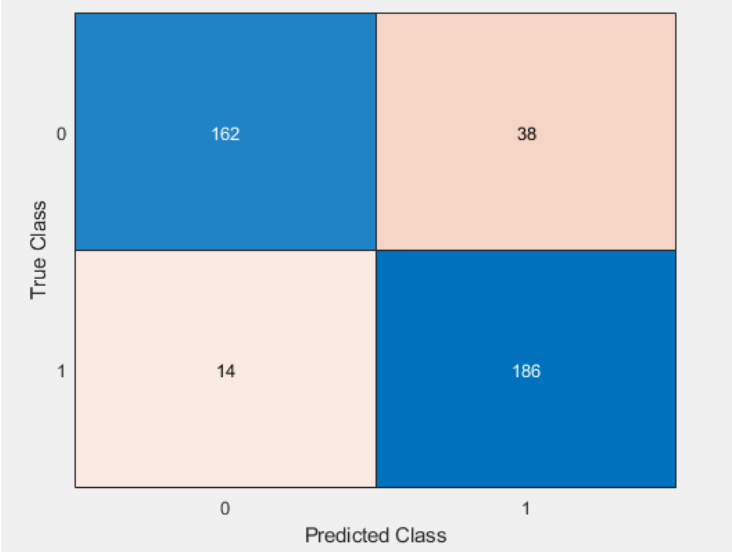
tp: 133, tn: 136, fp: 64, fn: 67, accuracy: 0.672500

precision: 0.68, recall: 0.67, specificity: 0.68, negative predictive value: 0.67

false positive rate: 0.32, false discovery rate: 0.32, f1: 0.67, f2: 0.67

Performance Metrics for mini batch Inception dataset

Training time: **14.2** seconds, Confusion Matrix:



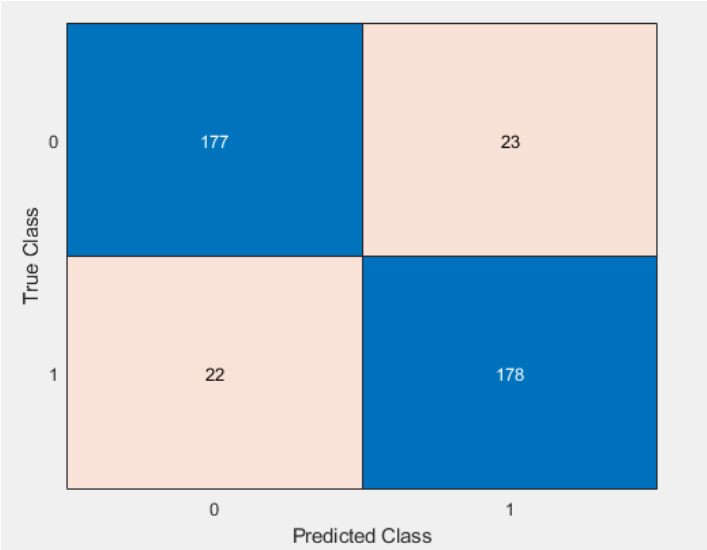
tp: 186, tn: 162, fp: 38, fn: 14, accuracy: 0.870000

precision: 0.83, recall: 0.93, specificity: 0.81, negative predictive value: 0.92

false positive rate: 0.19, false discovery rate: 0.17, f1: 0.88, f2: 0.91

Performance Metrics for full batch Inception dataset

Training time: **2.9** seconds, Confusion Matrix:



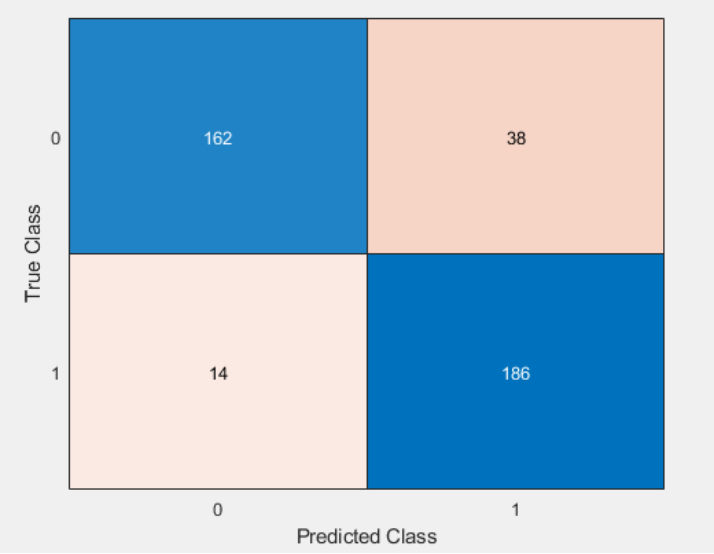
tp: 178, tn: 177, fp: 23, fn: 22, accuracy: 0.887500

precision: 0.89, recall: 0.89, specificity: 0.89, negative predictive value: 0.89

false positive rate: 0.12, false discovery rate: 0.11, f1: 0.89, f2: 0.89

Performance Metrics for stochastic Inception dataset

Training time: **39.5** seconds, Confusion Matrix:



tp: 186, tn: 162, fp: 38, fn: 14, accuracy: 0.870000

precision: 0.83, recall: 0.93, specificity: 0.81, negative predictive value: 0.92

false positive rate: 0.19, false discovery rate: 0.17, f1: 0.88, f2: 0.91

Comments:

- First of all, there were no distinctive performance result in comparison to 3 different algorithms in both Hog and inception dataset. I get similar results in all 3 approaches.
- The difference of algorithms came in to hand in regard to their runtime. Their order in terms of run times is

  Stochastic batch > mini batch > full batch

  However. It should be noted that this order could be different if I had different datasets for example a dataset with a lot of training samples. The reason for this note is that memory of the computer may not be enough to perform vectorized approach of the full batch.
- In comparison to inception and Hog dataset. It can be stated that inception dataset is relatively more linearly separable in comparison to Hog dataset. Inception dataset is a better feature extraction method to use in this model.
- For the performance metrics results I couldn't notice anything inconsistent between them such as high accuracy but low precision. I believe reason for this is our test data is distributed equally and our training model is consistent in terms of two different classes.
- For the part *Report 10 most important features (in terms of indices) and discuss the relation between weights and individual importance of features,* I didn't put the indices into report since but I printed them in the console of the code implementation. What I found in terms of importance of the features is when we have standardized dataset as in the homework, the features with higher (absolute value) weights are more important in Logistic Regression model because they effect decision boundary result more in comparison to other features.

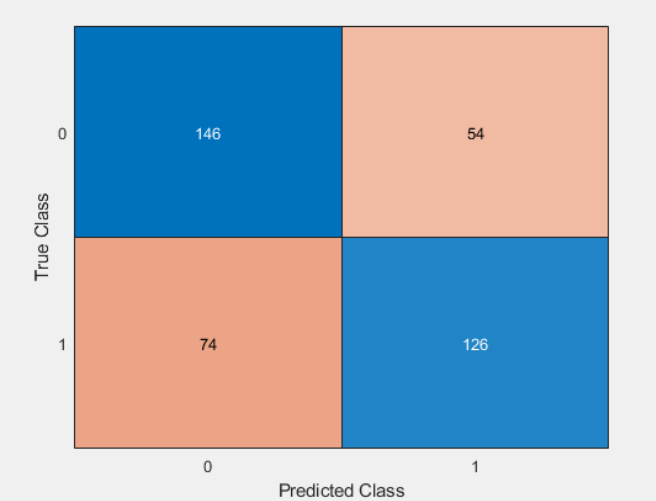
Question 1.4

Results for soft margin linear kernel cross validation for Hog dataset:

C:	10^-2	10^-1	1	10	100
Accuracy	0.66	0.67	0.66	0.65	0.63

Chosen C value: 10^-1, Performance Results:

Training time: **66.4** seconds, Confusion Matrix:



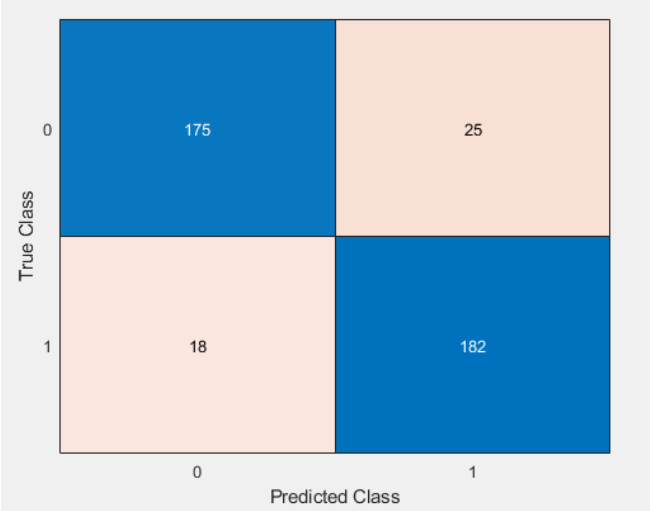
accuracy: 0.680000, precision: 0.70, recall: 0.63

Results for soft margin linear kernel cross validation for Inception dataset:

C:	10^-2	10^-1	1	10	100
Accuracy	0.88	0.85	0.85	0.85	0.85

Chosen C value: 10^-2, Performance Results:

Traingin time: **26.5** seconds, Confusion Matrix:



accuracy: 0.892500, precision: 0.88, recall: 0.91

Comments:

- Since linear kernel SVM is very similar to logistic regression I get similar results. I should state that even the soft margins of SVM approach couldn't improve the accuracy results that much in comparison to logistic regression.
- Inception dataset gave better performance metrics results in comparison to Hog dataset as earlier.
- I don't know why but one important thing to note is that Hog dataset implementation took longer than inception dataset implementation. Considering that inception dataset is bigger in terms of size of the data this was surprising to me.
- In terms of cross validation results for different C values there were nothing important to note. We couldn't see the importance of tuning C value in this experiment. There was no overfitting or underfitting according to my examination. Maybe if we tried different C values such as bigger values, we could get worse results due to high bias.

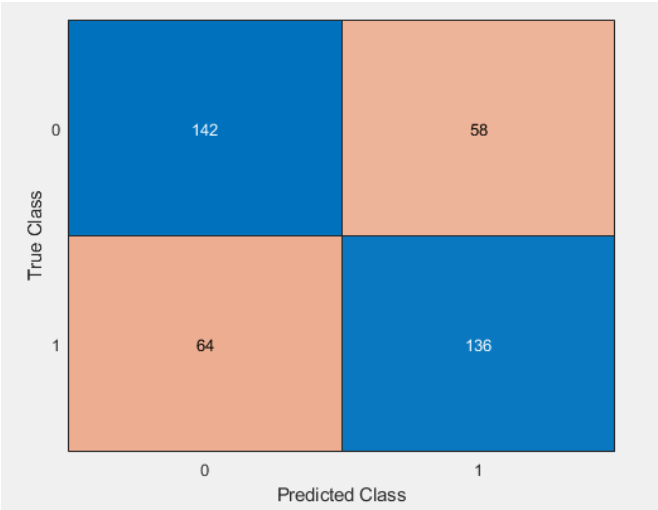
Question 1.5

Results for hard margin rbf kernel cross validation for Hog dataset:

Gamma:	2^-4	2^-3	2^-2	2^-1	1	2	64
Accuracy:	0.66	0.67	0.67	0.67	0.70	0.66	0.5

Chosen gamma value: 1, Performance Results:

Training time: **8.6** seconds, Confusion Matrix:



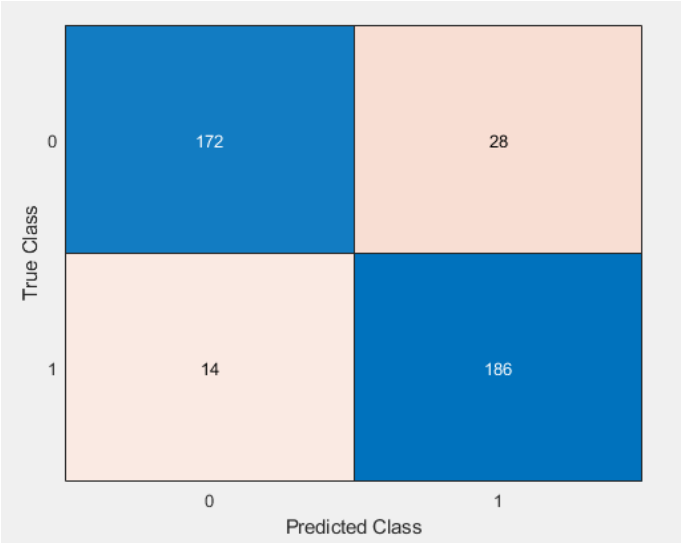
accuracy: 0.695000, precision: 0.70, recall: 0.68

Results for hard margin rbf kernel cross validation for inception dataset:

Gamma:	2^-4	2^-3	2^-2	2^-1	1	2	64
Accuracy:	0.88	0.87	0.57	0.50	0.50	0.50	0.50

Chosen gamma value: 2^-4, Performance Results:

Training time: **73.5** seconds, Confusion Matrix:



accuracy: 0.895000, precision: 0.87, recall: 0.93

Comments:

- I get similar accuracy results in comparison to the linear kernel approach. Nothing important to note.
- Run time comparisons of the training models are interesting. For Hog dataset hard margin rbf kernel was faster than linear kernel, whereas, for inception dataset the results were inverse. Thus, I can say that we cannot be sure that which approach is better in terms of runtimes. But we also need to consider in linear kernel we tried 5 different C values and in rbf kernel we tried 5 different gamma values.
- Inception dataset gave better performance metrics results in comparison to Hog dataset as earlier.
- This time training inception data took longer than Hog data. Considering that inception dataset is bigger in terms of size of the data this makes more sense than before.
- In terms of cross validation results for different gamma values there were some interesting results. As we can see smaller gamma values gave better results in both Hog and inception datasets, but their thresholds for bad results were different. I assume that when we use big gamma values to tune, the model underfits. Additionally, we can't exactly say that smaller gammas will always give better results as we can see in Hog dataset.

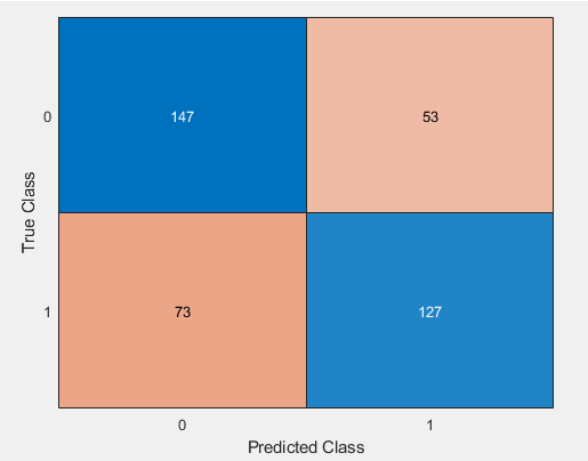
Question 1.6

Results for soft margin rbf kernel cross validation for Hog dataset:

Gamma\C	10^-2	1	100
2^-2	0.65	0.67	0.67
2	0.52	0.66	0.67
64	0.5	0.5	0.5

Chosen C value: 1 and gamma value: 2^-2, Performance Results:

Training time: **15.8** seconds, Confusion Matrix:



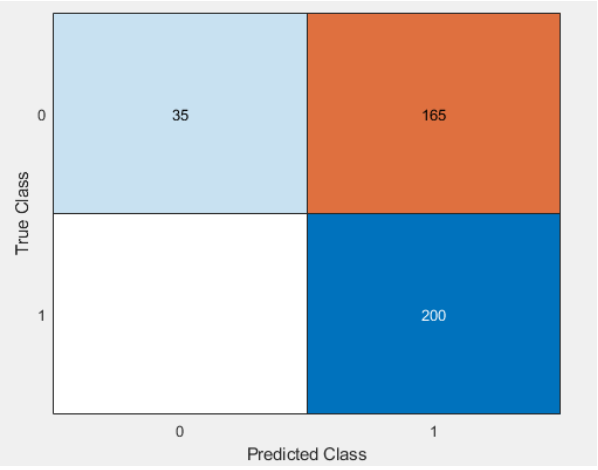
accuracy: 0.685000, precision: 0.71, recall: 0.64

Results for soft margin rbf kernel cross validation for inception dataset:

Gamma\C	10^-2	1	100
2^-2	0.51	0.57	0.58
2	0.53	0.53	0.5
64	0.5	0.5	0.5

Chosen C value: 100 and gamma value: 2^-2, Performance Results:

Training time: **99.1** seconds, Confusion Matrix:



accuracy: 0.587500, precision: 0.55, recall: 1.00

Comments:

- This time accuracy results were interesting. Unlike other models, this time I get relatively bad results for inception dataset.
- Run times for this training approach are bigger than earlier implementations. I believe that this is mostly because this time we tried 9 different combinations for our C and gamma combinations.
- According to my observations, lower C and bigger gamma leads to underfitting the data or high bias.
- In terms of cross validation results for different gamma values there were some interesting results again. As we can see when gamma is big, sometimes even C value cannot help to avoid underfitting the data. Similarly, we can say the reverse when C is smaller, sometimes gamma value cannot help for better results. But these derivations are according to the results I observed.
- From all of the combinations of C and gamma, I could not find a good combination to train inception data with soft margin rbf kernel approach.
- One other thing to note that, although my accuracy is %58 looking at my recall which is %100, I can say that something is wrong with my model. I highly bias over predicting positive.

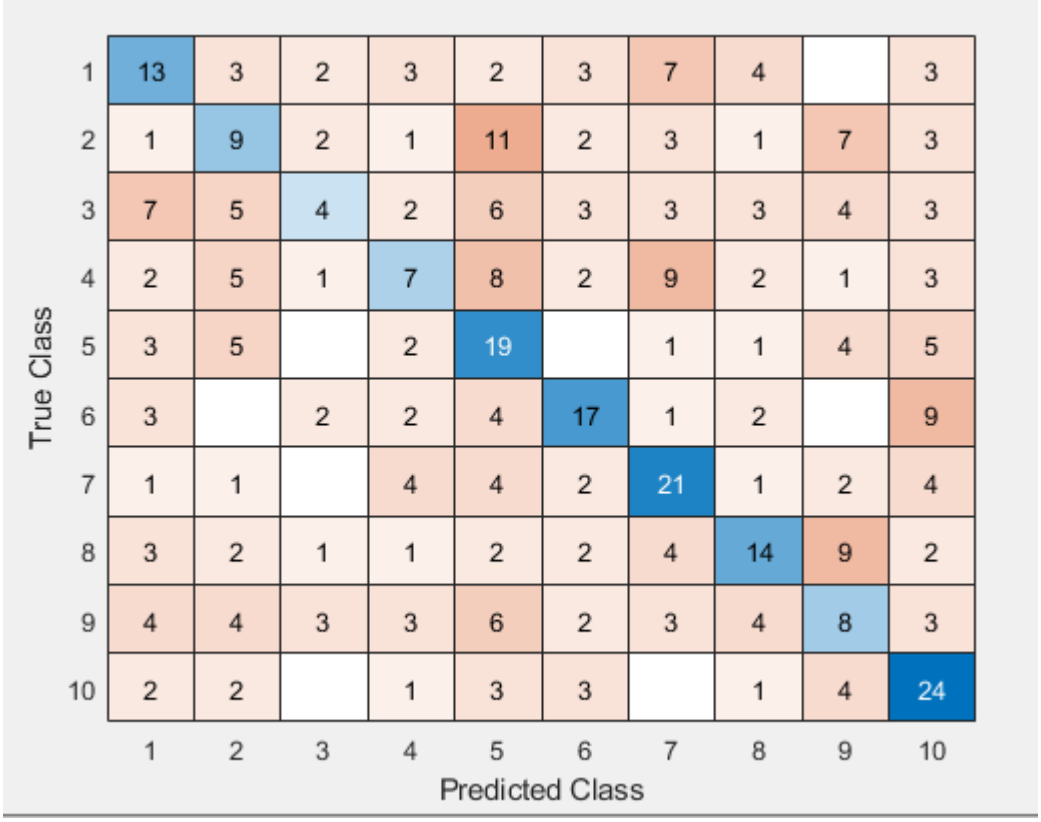
Question 1.7

Results for soft margin rbf kernel and one-vs-all cross validation for Hog dataset:

Gamma\C	10^-2	1	100
2^-2	0.12	0.12	0.11
2	0.33	0.34	0.32
64	0.27	0.25	0.28

Chosen C value: 1 and gamma value: 2, Performance Results:

Training time: **110.9** seconds, Confusion Matrix:



Class	1	2	3	4	5	6	7	8	9	10
Accuracy:	0.32	0.22	0.1	0.17	0.47	0.42	0.52	0.35	0.2	0.6

macro recall: 0.340000, macro precision: 0.332376, macro f1: 0.336145

micro recall: 0.340000, micro precision: 0.340000, micro f1: 0.340000

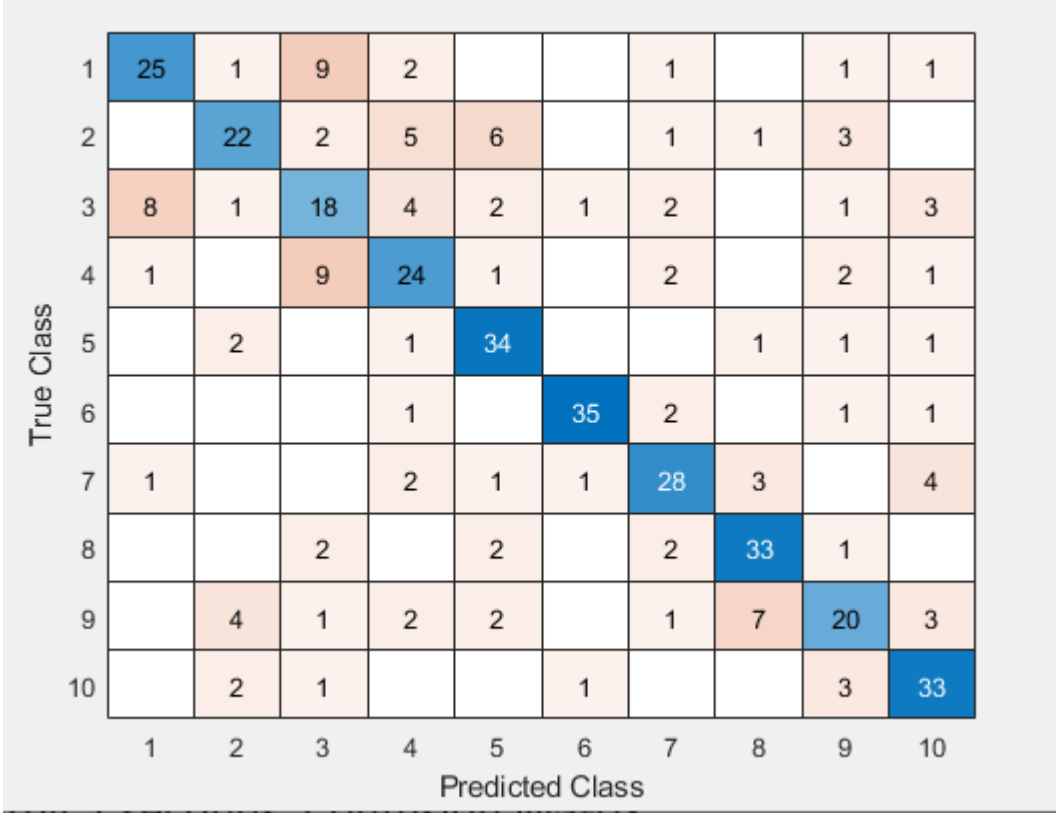
total accuracy: 0.34

Results for soft margin rbf kernel, one-vs-all cross validation of inception dataset:

Gamma\C	10^-2	1	100
2^-2	0.1	0.1	0.1
2	0.13	0.13	0.12
64	0.62	0.64	0.66

Chosen C value: 100 and gamma value: 64, Performance Results:

Training time: **764.7** seconds, Confusion Matrix:



Class	1	2	3	4	5	6	7	8	9	10
Accuracy:	0.62	0.55	0.45	0.60	0.85	0.87	0.70	0.82	0.5	0.82

macro recall: 0.680000, macro precision: 0.680458, macro f1: 0.680229

micro recall: 0.680000, micro precision: 0.680000, micro f1: 0.680000

total accuracy: 0.6800

Comments:

- I won't compare these results with earlier results since this time we are looking for subclass labels.
- The trend of having better results with inception datasets still continues.
- Considering class-based accuracies, inception datasets is better distributed. The difference between best and worst class accuracy of Hog dataset is %50, whereas the difference best and worst class accuracy of inception data set is %42.
- Considering parameters for Hog dataset, I don't know whether it is because we didn't try good parameters or dataset is not suitable for better results, we don't have good results in general. However, there are no favorable sides in both increasing or decreasing gamma or C value.
- Considering parameters for inception dataset, we can say that we have better results compared to Hog dataset, but not with all parameter combinations. It is very clear that, when we have small gamma, we cannot reach good accuracies. Maybe this is because we overfit the data with small gamma values. I couldn't see any distinctive effect of C value in my observations.
- Although they are very big, since we used one vs all approach this much training times are expected.
- Another thing to mention is all of macro micro precision and recall values and accuracy are very close to each other. I believe this is because our test dataset is equally distributed and when predicting we don't heavily favor one class.

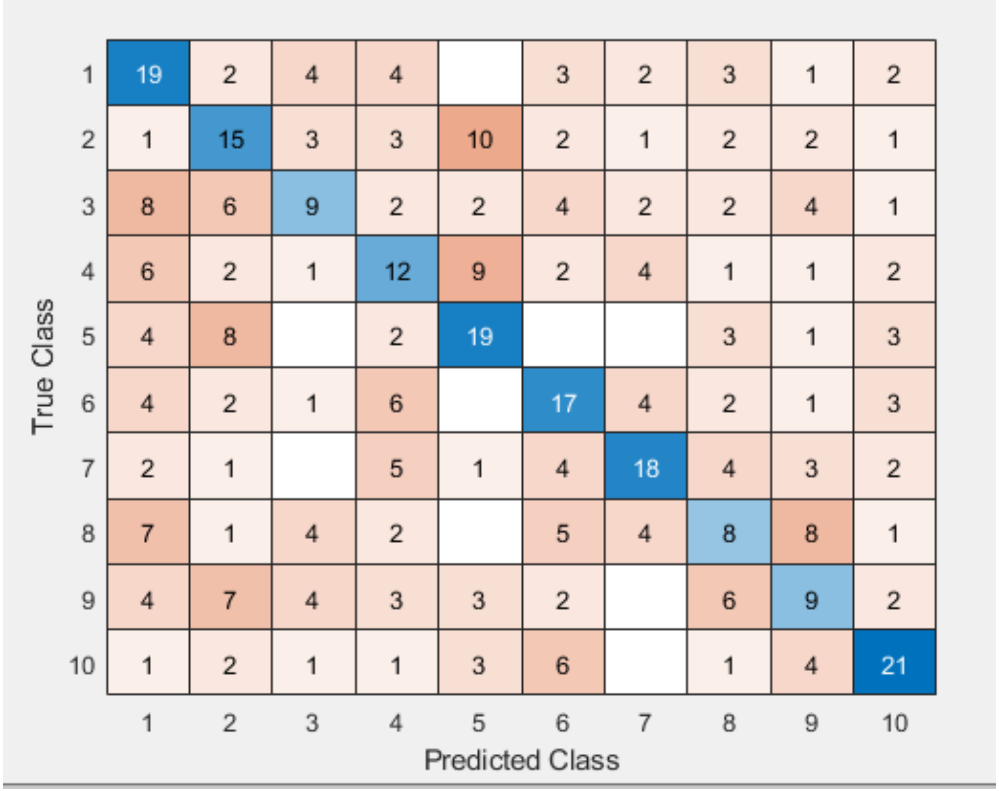
Question 1.8

Results for hard margin polynomial kernel cross validation for Hog dataset:

Gamma\Degree	3	5	7
2^-2	0.32	0.34	0.34
2	0.31	0.32	0.33
64	0.19	0.19	0.19

Chosen degree: 7 and gamma value: 2^-2, Performance Results:

Training time: **144.1** seconds, Confusion Matrix:



Class	1	2	3	4	5	6	7	8	9	10
Accuracy:	0.47	0.37	0.22	0.30	0.47	0.42	0.45	0.20	0.22	0.52

macro recall: 0.367500, macro precision: 0.366236, macro f1: 0.366867

micro recall: 0.367500, micro precision: 0.367500, micro f1: 0.367500

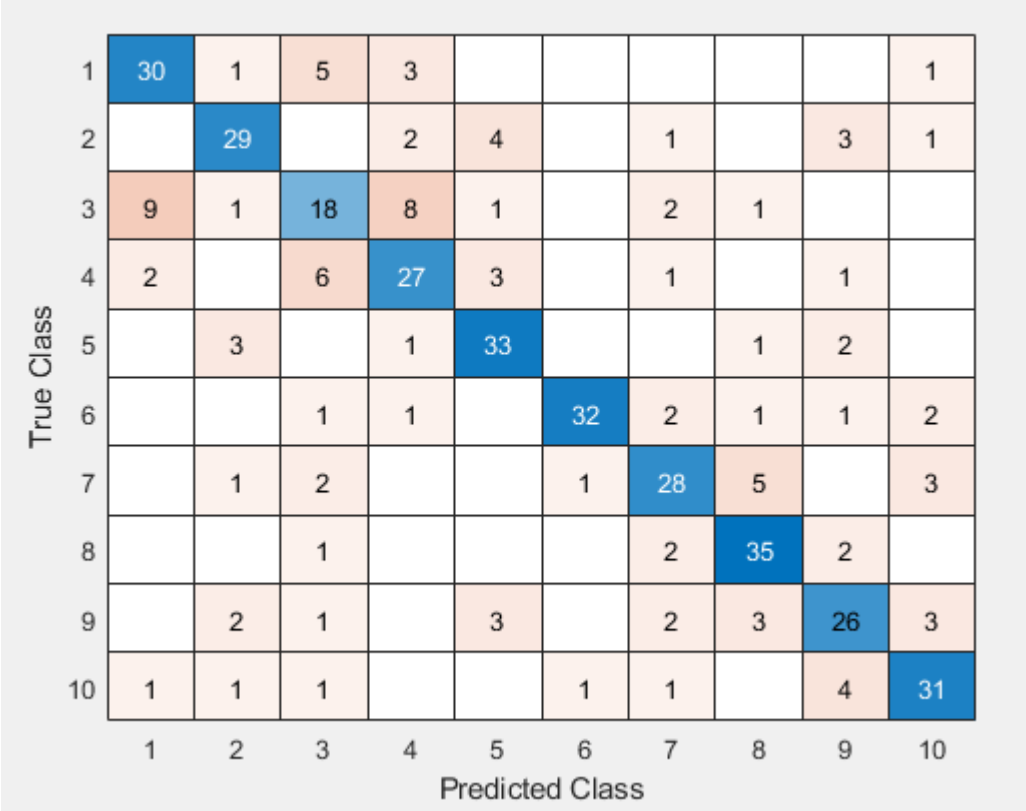
total accuracy: 0.3675

Results for hard margin polynomial kernel cross validation for inception dataset:

Gamma\Degree	3	5	7
2^-2	0.666	0.1	0.1
2	0.668	0.661	0.11
64	0.63	0.64	0.663

Chosen degree: 3 and gamma value: 2, Performance Results:

Training time: **112.7** seconds, Confusion Matrix:



Class	1	2	3	4	5	6	7	8	9	10
Accuracy:	0.75	0.72	0.45	0.67	0.82	0.80	0.70	0.87	0.65	0.77

macro recall: 0.722500, macro precision: 0.722735, macro f1: 0.722617

micro recall: 0.722500, micro precision: 0.722500, micro f1: 0.722500

total accuracy: 0.7225

Comments:

- Compared to soft margin rbf kernel results the accuracy results are both slightly better. I don't think that this means polynomial kernel is always better than rbf kernel, rather, I believe that our dataset and chosen parameters to try fits better to polynomial kernel.
- Considering runtime of rbf kernel there is a huge decrease in polynomial kernel but just with inception dataset not with Hog dataset. Moreover, Hog dataset polynomial kernel training lasts longer than inception dataset. Considering that inception dataset is bigger in terms of size of the data this was surprising to me.
- I believe class-based accuracy distributions are slightly better in both datasets compared to soft margin rbf kernel approach.
- Considering parameters for Hog dataset, bigger degree and smaller gamma tends to give better results. I believe the opposite combinations of parameters tends to underfit the data.
- Considering parameters for inception dataset, I can say that there are surprising results. Unlike my expectations, smaller degree gives better results for inception dataset. I believe higher degree polynomial kernels overfits data.
- Similar to rbf kernel subclass training, macro micro precision and recall values and accuracy are very close to each other. Again, I believe this is because our test dataset is equally distributed and when predicting we don't heavily favor one class.

Question 1.9

- Although there are few exceptions, I can say that transfer learning with Inception v3 feature extraction is better in terms of performance matrix results.
- For superclass my best results were when I used hard margin radial basis function kernel for inception dataset. However, there were other methods that yield very close results.
  For superclass my worst results were when I used soft margin radial basis function kernel for inception dataset. I believe that, this is not because using soft margin radial function kernel or inception dataset is bad. I think, the parameters that I tried couldn't prevent model from underfitting.
  For subclass my best results were when I used hard margin polynomial kernel for inception dataset. I believe this approach and inception data fits well together in comparison to other models.
  For subclass my worst results were when I used soft margin rbf kernel and one-vs-all for Hog dataset.
- Considering my observations, I would say that when training SVM if C value is relatively small training results have high bias or underfit, if C value is relatively big training results have high variance or overfit.
- Considering my observations, I would say that when training SVM with RBF kernel, if gamma value is relatively big training results have high bias or underfit, if gamma value is relatively smaller training results have high variance or overfit.
- Considering my observations, I would say that when training SVM with polynomial kernel, if degree is relatively small training results have high bias or underfit, if degree is relatively big training results have high variance or overfit.
- Considering my observations, I would say that when training SVM with C and gamma parameters, if C is relatively small and gamma is relatively big training results have high bias or underfit, if C is relatively big and gamma is relatively small training results have high variance or overfit.
- Considering my observations, I would say that when training SVM with polynomial kernel with C and gamma parameters, if degree is relatively small and gamma is relatively big training results have high bias or underfit,

if degree is relatively big and gamma is relatively small training results have high variance or overfit.

- When batch size increase training time of logistic regression decreases because of vectorized approach of gradient ascent. However, this conclusion is valid when there is enough memory as computer hardware.
- Considering my observations, I would say that when training SVM with polynomial kernel, if gamma value is relatively big training results have high bias or underfit, if gamma value is relatively smaller training results have high variance or overfit.
- Logistic regression is relatively easier to implement and runtime is smaller since we don't have to tune any parameters. On the other hand, although implementation of SVM is harder and longer, if rights parameters can be found its performance metrics tends to give better results than logistic regression because of its soft margin approach.

Question 1.10

- Negative predicted value is important when we want to be sure that no positive class should be overlook. For example, right now we have covid-19 crisis. NPV value is important for covid-19 situation since detecting all positive cases are important, we want to be sure that if the test gives negative result it should be reassuring. When NPV is higher it is more reassuring.
- Again, in covid-19 context, false positive rate is important when we want to be sure that no resource should be allocated to wrong people. Meaning that, if some person detected as sick it shouldn't be actually healthy. We want false positive rate to be small.
- We want false discovery rate to be small when in general we don't have tolerance to wrong results.
- F1 score is important when we have recall or precision inconsistent with accuracy. For example, I have a situation when using soft margin rbf kernel for inception dataset.
  My results were accuracy: 0.587500, precision: 0.55, recall: 1.00.
  In this case F1 score would be helpful for me to think that %58 accuracy has some anomalies.

Question 2.1

It took **1.2** seconds to implement SVD based PCA to dataset.
If we say X is m x n matrix

centerX is X – column means of X and
centerX = USV^T is the singular value decomposition of centerX.

There are two ways to reconstruct the data. We can use U x S or centerX x V as principal component scores and reconstruct the data by applying

PCscores*V^T + column means

When I used U x S for PCscores, the MSE was 5.0094009e-08
When I used centerX x V for PCscores, the MSE was 3.2220434e-08

I can say that both results can be ignored since they are very small and probably resulting from MATLAB. We can conclude that SVD reconstruction is very consistent with original data.

Question 2.2

It took **11.2** seconds to implement covariance matrix based PCA to dataset.
Similar to question 2.1 if we say

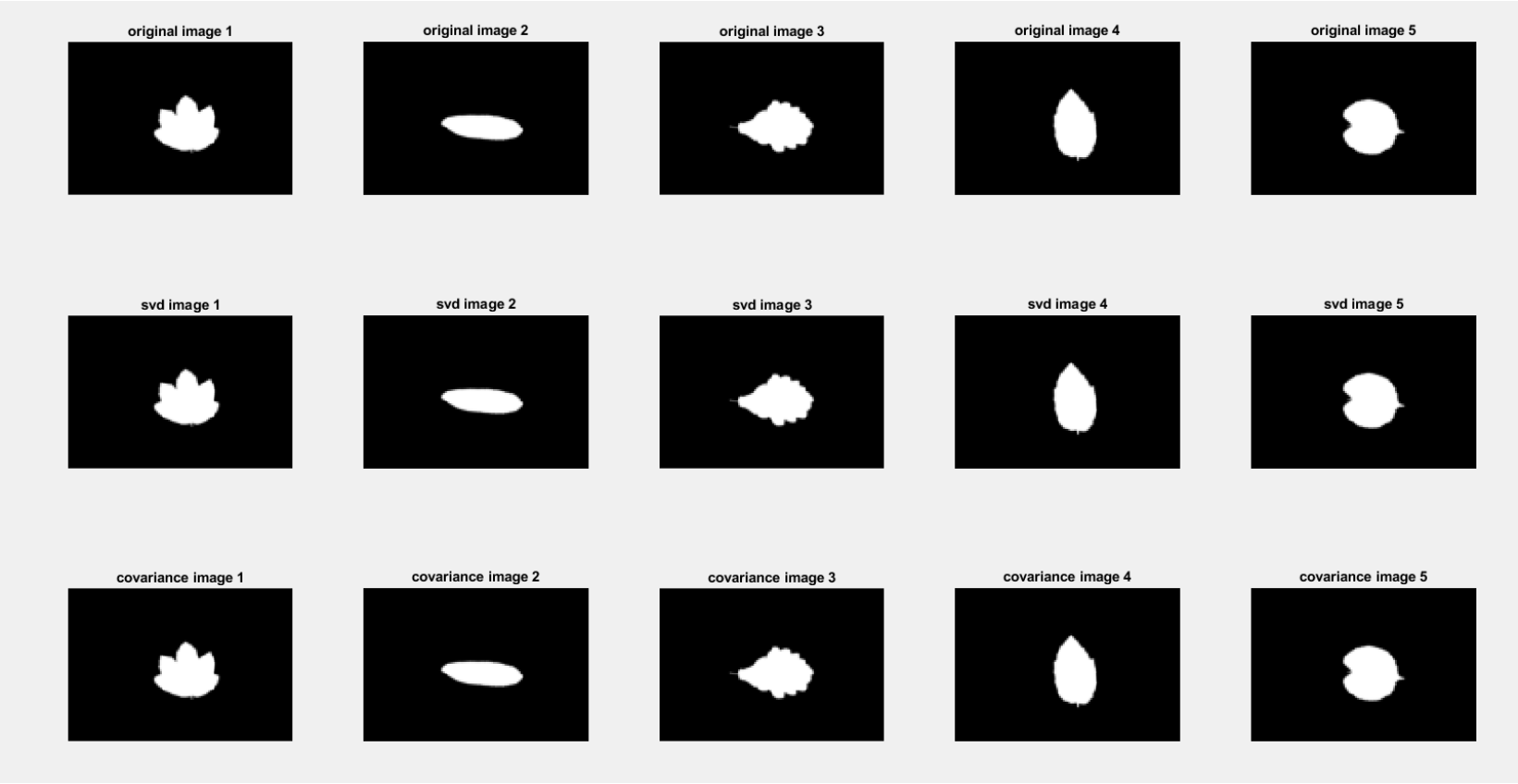
Covariance matrix = $centerX^T$ x centerX / m and
Covariance matrix = VLV^T

We can use centerX x V as principal component scores and reconstruct data with

PCscores*V^T + column means

The MSE I get was 2.4689191e-07. Although this is slightly bigger than SVD results, it is still can be ignored. It would be good to point out that covariance implementation lasts dramatically longer than SVD approach.

Question 2.3



It is very easy to state that SVD approach is better than covariance matrix approach considering runtime of algorithms as I stated their runtimes above in question 2.1 and 2.2. For my experience MSE values was very small and difference of the images above can not be detected by me. Thus, I can state that in this case MSE values can be reliable measure.

I can't think of any improvement to MSE in this context.

Question 2.4

In order to answer this question, I assumed that we use covariance matrix for pca and I also assumed,

when (i) $n_{samples} >> n_{features}$,
$n_{samples} = 3k$ and $n_{features} = k$

when (iii) $n_{samples} << n_{features}$,
$n_{samples} = k$ and $n_{features} = 3k$

In this case. Since the size of covariance matrix is $n_{features} \times n_{features}$
the first one will have k x k covariance matrix and
the second one will have 3k x 3k covariance matrix. Since we need to find eigen values of covariance matrix smaller covariance matrix size is preferable.

In this case I would say running time comparison would be **(iii) > (i).**

In order to include (ii) $n_{samples} \sim n_{features}$ into comparison, I believe that more information needs to be given.

If we say that
when (ii) $n_{samples} \sim n_{features}$,
$n_{samples} = k$ and $n_{features} = k$

Then running times would be **(iii) > (i) > (ii)**

If we say that
when (ii) $n_{samples} \sim n_{features}$,
$n_{samples} = 3k$ and $n_{features} = 3k$

Then running times would be **(ii) > (iii) > (i)**